Laboratory Assignment 3

September 29, 2015

Austin Huck

Taylor Okel

# Objectives and Problem Description

Part I – Conveyor Belt

Create a simulated conveyor belt. Boxes will arrive in front of a photodetector in four sizes – 10, 20, 30, or 40 inches. The conveyor itself will move at 10 inches per second. The program should detect boxes as they come across, recording the various sizes. Boxes will arrive in more than 10 inches, but no more than 1000 inches. In the case of a gap of more than 1000 inches, the system should alert the user of a jam via blinking LED and debug messages and increment a jam counter. The user will be able to clear this state through the use of a pushbutton. The data gathered by the program, box and jam counts, will be stored in EEPROM, such that the next operation will be able to continue the counting process and display the previous circuit’s data on start up. Shifts will last eight hours, after the end of which the conveyor and counting system will cease operation and display the counts of the shift. During the entire shift, the system will display a live graph of the total number of boxes received as a function of time.

Part II – Mars Rover

Create a light level recorder for a Mars rover. The detected light level must be recorded every 15 minutes using a photodetector. Every 24 hours the data must be transmitted back to Earth. Due to the limited power of the solar/battery setup this must be as efficient as possible.

# Procedure

Both portions of this lab will utilize the photo detector, while part I requires an LED and pushbutton to operate the jam sequence. Thus, one design can be utilized for both circuits, as shown in Figure 1. The component utilization and pin assignments are tabulated in tables one and two.

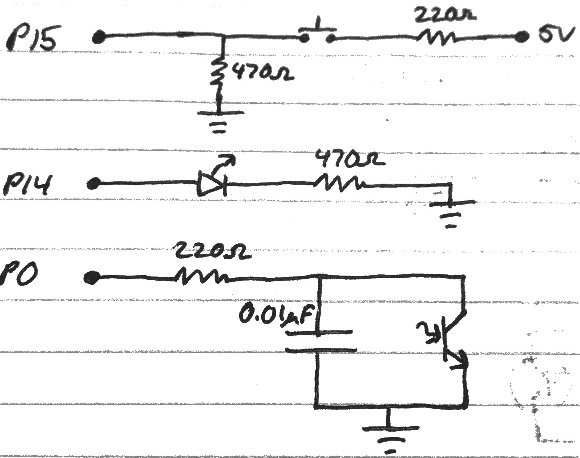


Figure 1 - Conveyor Belt & Mars Rover Circuits

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Components | | | | |
| Resistor - 220Ω (2) | Resistor – 470 Ω (2) | Photodetector (1) | Capacitor - 0.1uF (1) | LED (1) |

Table 1 - Full Component Values

|  |  |  |
| --- | --- | --- |
| Pins | | |
| 0 (in) Photodetector | 14 (out) LED [Conveyor Only] | 15 (in) Button [Conveyor Only] |

Table 2 - Full Circuit Pins

Part I – Conveyor Belt

The conveyor belt will need to begin by loading the data in EEPROM and displaying it to the user, then initializing a graph to display the box count. It then moves into the operating state, where it will wait for a box detection. Once a box is detected, the system will measure the box via photodetector. Since the conveyor moves at a rate of 10 inches per second, each second will denote the next size of box. The count for the determined box will be incremented, and the EEPROM will be updated with the new value. In the case of a jam (no detections or a detection lasting longer than 1000 inches, or 100 seconds), a jam sequence is entered. This sequence alerts the user via debug message and blinking LED, as well as increments the jam count. The user can clear the jam state via pushbutton. The shifts last eight hours; using a timer, the system will automatically end at this point, display the data, and exit. This logic was omitted from the original flowchart (figure 3), so is appended in figure 4. The check occurs before the “Box Detected” check, where the black box “System” in figure 4 is the main portion of the program as seen in figure 3.

To run the photodetector, we sensed ambient light at a value of about 100, and decided on a good maximum of about 1000 by placing a hand over the detector. Using these values, the transition states were determined via the following:

Equation 1 - "No Box" Range

Equation 2 - "Box" Range

Which gave us transition states for the box, as well as a non-transition region to prevent error. Figure 2 gives a visual of the transitions.

Box

Non-Transition

No Box

Figure - Detection Ranges

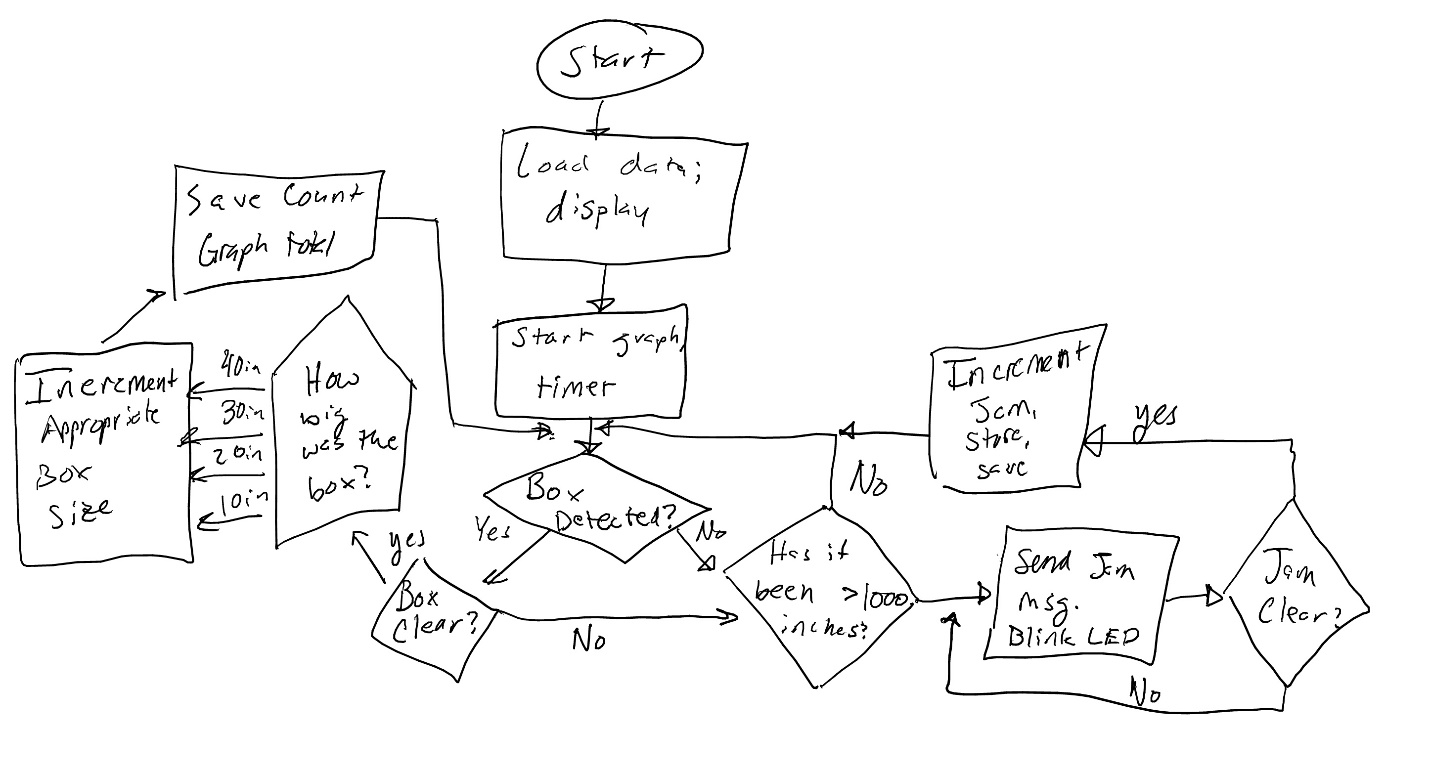


Figure 3 - Conveyor Belt Flowchart

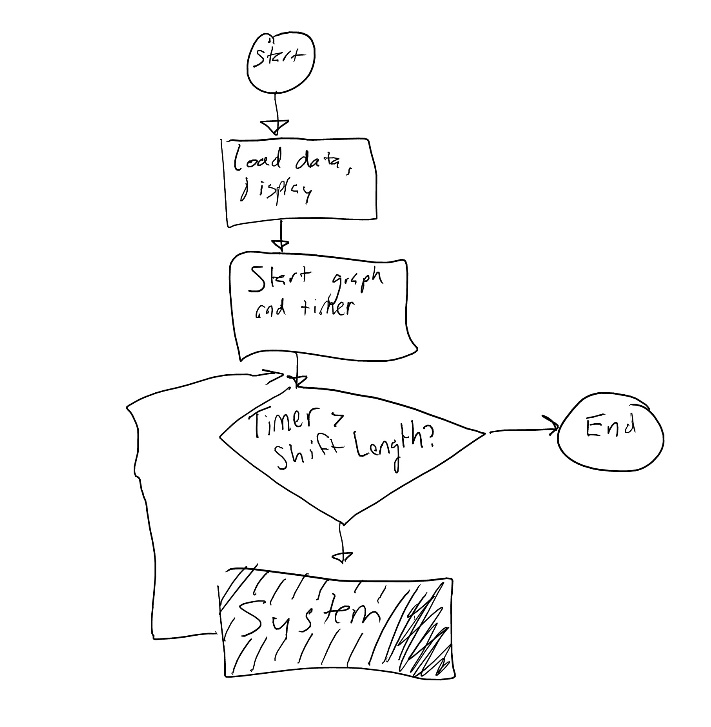


Figure 4 - Conveyor Belt Flow Addition

Part II – Mars Rover

The light recording routine will begin with recording the first light level. Using RCTIME and the circuit on pin 0 of figure 1, this will be a value between 0 for brightest and about 20000 for darkest. The controller will proceed to wait for 15 minutes in a low power state using the SLEEP command with parameter 900 (seconds). After 96 light level samples have been taken (24 hours / 15 minutes = 96) the program will enter the transmit routine where all data will be sent. In this prototype the data will be sent via debug serial connection to a connected computer. The EEPROM data index will be reset to 0 and data collection will proceed at 15 minute intervals.

SLEEP is the most accurate and power efficient way of waiting for 15 minutes with the basic stamp 2. According to documentation, SLEEP will result in a current draw of about 50µA for a total of 1.2mAh required battery capacity for daily sleep time. Account for an additional 0.1mAh for daily light level collection. SLEEP also has an accuracy of about ±1% @ 75°F.

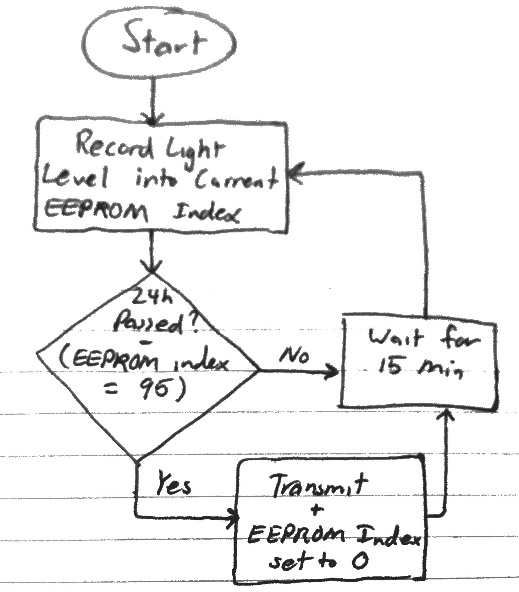


Figure 5 - Mars Rover Flowchart

# Expected Results

Part I – Conveyor

The system will begin by displaying the previous run’s data as well as a graph of today’s box count. The program will then wait for a box to pass the sensor. When a box does so, the system measures the size and records the data both locally and in the EEPROM (the graph will also show an increased count). If no boxes appear for 100 seconds, the jam state will occur with a blinking light. Hitting the pushbutton for more than 1 second will clear this state, increment the jam count, and store the jam count and duration in the EEPROM. Once the shift is over, the program will end and display the box and jam counts to date.

Part II – Mars Rover

Light level collection will begin immediately and occur every configured time interval. For testing the collection interval was set to 100ms in order to execute the transmission logic every 10 seconds. Transmitted data should be in the form of text formatted as 00:00 00000 for time and value delimited by carriage return for each sample. When adjusting the collection interval to 15 minutes it should be observed to take samples every 15 minutes, entering a low power state in between time, and send the data every 24 hours.

# Experiment and Design Revisions

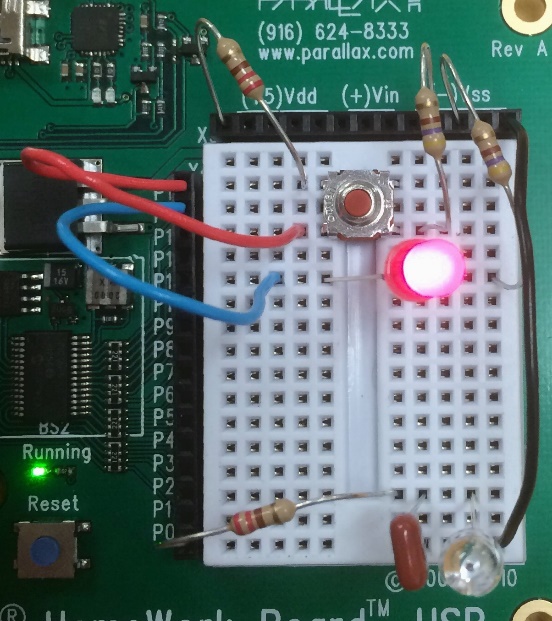


Figure 6 - Completed Circuit

Part I – Conveyor

The system had a few minor bugs which were ironed out. Examples of these included the jam sequence with the LED not functioning properly, issues writing the box count to EEPROM (countBox is an array), and various issues with box detection and appropriate timings to properly simulate the situation. After fixing these issues, the system worked as intended as stated by the objective statement.

Part II – Mars Rover

The light collection and transmission logic worked as expected upon first test. The SLEEP command was not initially utilized and was implemented in place of PAUSE to meet the power requirements. The transmitted data was also formatted to include the sample time. The SLEEP interval of 15 minutes was accurate within the indicated ±1%. There are various afterthought revisions that could be made to this project. The light level values are in terms of RCTIME’s output. Therefore a greater value indicates a darker light level. This is not exactly intuitive. An excellent revision would be to translate this RCTIME value to an arbitrary unit of brightness or lumens. Depending on required accuracy and environmental factors, a more accurate timing mechanism may be necessary. If very dark light levels are expected, it would be necessary to revise the circuit to prevent RCTIME’s greater than the maximum word integer. Something such as a large value resistor in parallel with the photodetector would solve this.

# Observations

The most noticeably observation between both parts was the photodetector’s sensitivity. This allowed for detection of a box given constant background light in part 1. In part 2, the sensitivity made for fine sampling of the currently ambient light. Due to the requirement of constant level background light for part 1 this made the photodetector less than ideal. In a real system it cannot be assumed that the ambient light will always be constant. This also made edge detection of boxes fuzzy and therefore box length measurements have a limited accuracy. The photodetector was ideal for part 2 in the detection of ambient light levels. Theoretically the RCTIME’s measurement can be used to calculate a specific unit of brightness given circuit and environmental characteristics. An additional observation is that the timing capabilities of the BS2 has limited accuracy. Temperature changes must be taken into consideration when accurate timing is required. Depending on the circuit components, temperature may play a part on circuit responses as well.

# Discussion

This lab introduced more complex PBASIC programs and photodetectors. Advanced task requirements and the use EEPROM made for more planning and testing. A few key learning points include Basic Stamp EEPROM and photodetector usage. The PBASIC commands to use device EEPROM were used in both parts of this lab and introduced programming with non-volatile memory. Photodetectors were also used in both parts of the lab. Some applications of photodetectors, including object detection and light intensity recording, were introduced. Understanding how a photodetector behaves in a circuit was also fundamental to writing the PBASIC code to interpret the circuit response.

# Exercises

Photoreceptors detect light that is emitted out of a source *or* reflected off of a surface. The physics of light is such that specific colors absorb specific wavelengths, and reflect others. In other words, if you were to flood the area with a red light, the red boxes would not reflect any more or less light. Conversely, blue or green boxes would reflect or absorb differing amounts of specific wavelengths. Thus, we can detect differing box colors by sending boxes through several areas flooded with differing colors, and detecting changes in wavelengths. This data can be compiled at the end to make a decision on box color.

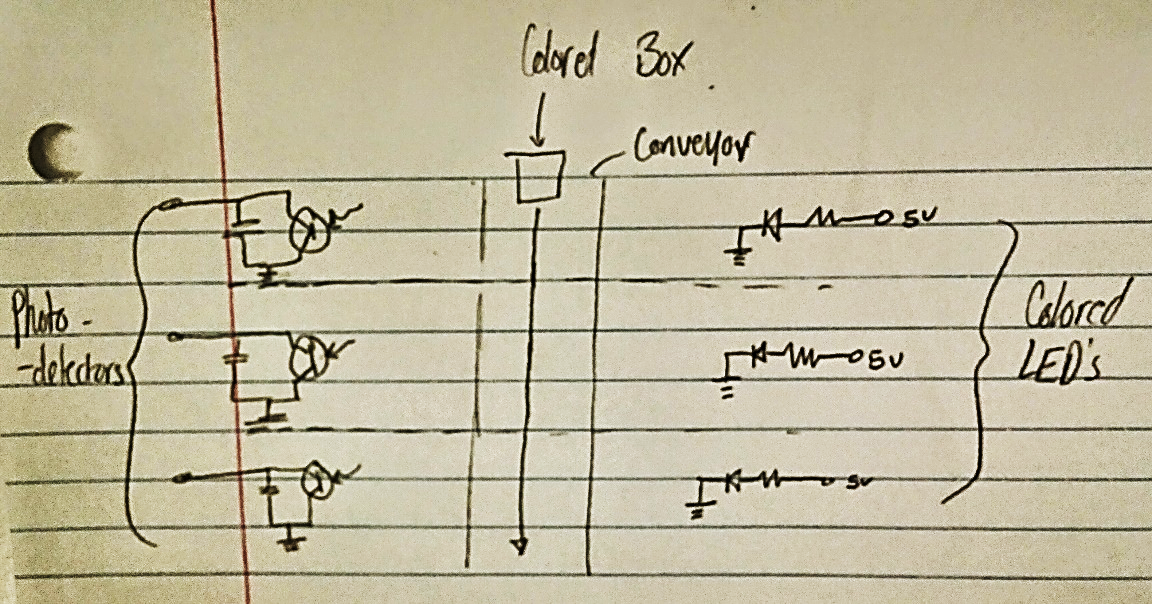


Figure 7 - Exercise Schematic